ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN
KHOA KỸ THUẬT MÁY TÍNH

THIẾT KẾ HỆ THỐNG NHÚNG

MINI PROJECT:
SINGLE PING-PONG GAME

HỌ VÀ TÊN: NGUYỄN GIA BẢO NGỌC – 21520366
NGUYỄN QUỐC TRƯỜNG AN – 21521810
NGUYỄN ĐỨC LƯU – 21522314

LỚP: CE224.O12

GIẢNG VIÊN HƯỚNG DẪN:
CHUNG QUANG KHÁNH

TP. HỒ CHÍ MINH – Tháng 1 năm 2024

MỤC LỤC:

Danh mục ảnh:

Danh mục bảng:

MINI PROJECT:
SINGLE PING-PONG GAME

1 CHUẨN BỊ

1.1 Phần cứng

- KIT STM32F4 Discovery for STM32F429 MCU.

1.2 Phần mềm

- STM32CubeIDE: sử dụng để lập trình, build, nạp và debug code.

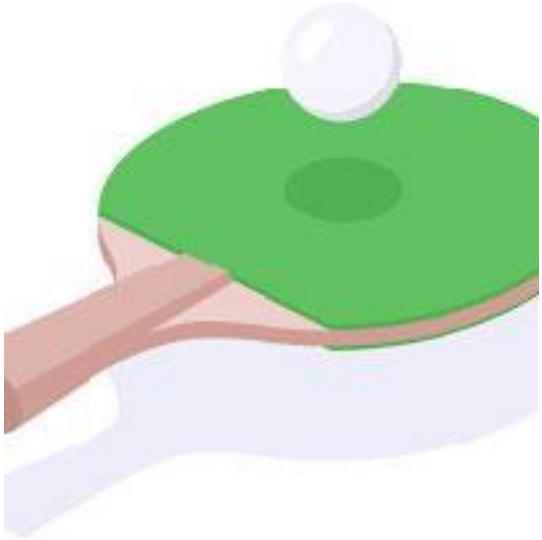
1.3 Kiến thức

- Biết cách tạo project và cấu hình project sử dụng STM32CubeIDE
- Có kiến thức vững về Hệ điều hành, đặc biệt là vấn đề lập lịch và đồng bộ các tiến trình

Hình 1 - KIT STM32F4 Discovery

2 MÔ TẢ

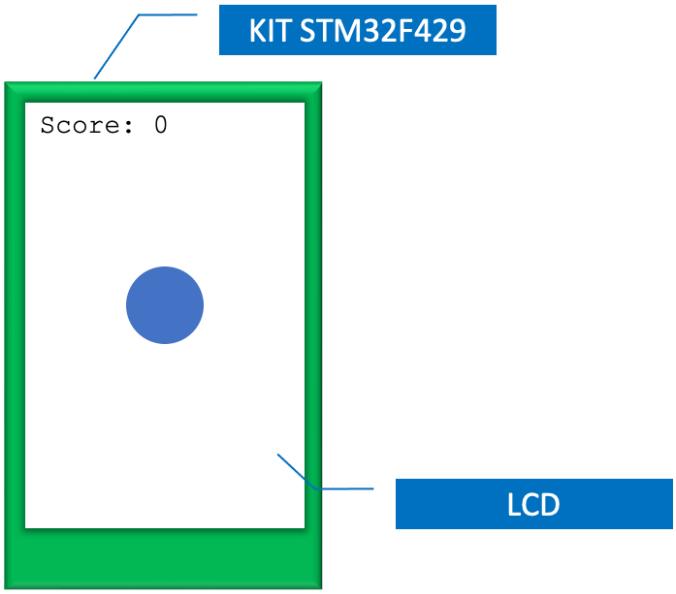
Single Ping-Pong Game là một trò chơi mô phỏng việc dùng vợt và tâng bóng bàn được mô tả như trong Hình 1, trong đó, KIT STM32F429 được sử dụng như cây vợt, màn hình LCD sẽ hiển thị một hình tròn mô phỏng trái bóng bàn.



Hình 2 - Mô tả game tâng bóng bàn

Nguyên lý của game như sau:

- Đầu tiên, hệ thống ở trạng thái cân bằng khởi đầu.
- Người chơi sẽ hạ và nâng KIT để thực hiện động tác tâng bóng, khi đó trên màn hình LCD sẽ mô phỏng trạng thái trái bóng được nâng lên và rơi xuống.
- Mục tiêu của người chơi là phải nâng KIT đúng lúc trái bóng rơi xuống, đập vào "mặt vợt" và tâng lên lại, người chơi được tính điểm, đèn xanh chớp 1 lần.
- Nếu người chơi nâng đúng, tùy thuộc vào mức độ chính xác mà bóng sẽ nảy lên với tốc độ khác nhau.
- Nếu người tâng "hụt" thì bóng sẽ rớt và GAME OVER, đèn đỏ được bật và giữ cho tới khi reset.



Trạng thái ở hệ thống
cân bằng ban đầu

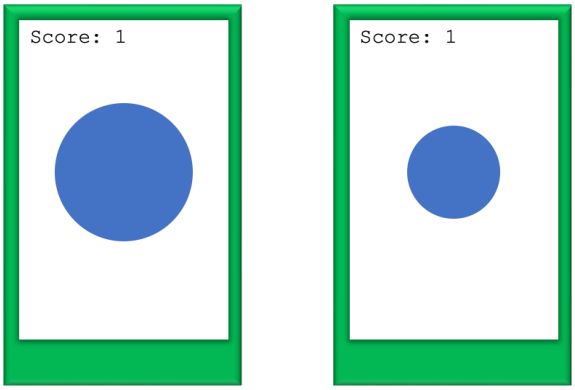
Hình 3 - Trạng thái bắt đầu của game

Động tác tâng bóng

Hạ KIT xuống

Nâng KIT lên để tâng bóng

Hình 4 - Mô phỏng động tác tâng bóng



Mô phỏng bóng bay lên Mô phỏng bóng rớt xuống

Hình 5 - Mô phỏng bóng được nâng và rớt

Yêu cầu về mức độ hoàn thành:

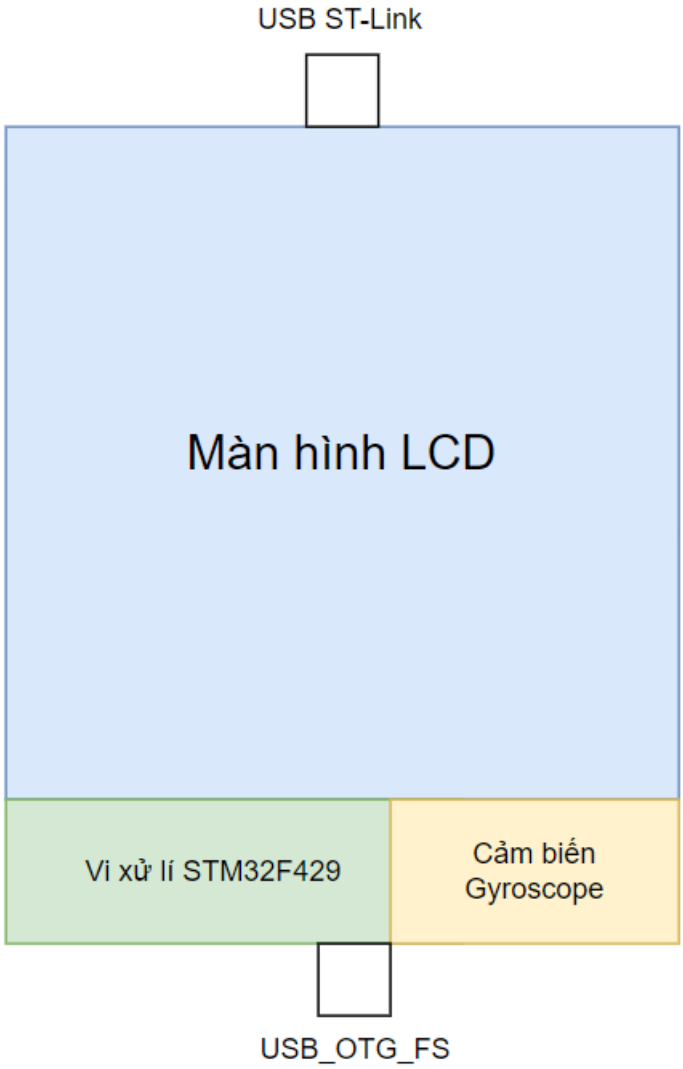
Bảng 1 - Các yêu cầu

Yêu cầu	Trạng thái
Hiển thị được giao diện ban đầu	Đã hoàn thành
Hiện thực được thao tác tâng bóng tại chỗ (điểm giữa màn hình)	Đã hoàn thành
Xuất thông số độ cao khi bóng được tâng ra máy tính qua Virtual Com Port	Đã hoàn thành
Hiện thực được chức năng tính điểm	Đã hoàn thành
Hiện thực được việc bóng bay theo 1 chiều dựa trên phương và chiều của lúc nâng bóng (nếu bóng bay ra rìa LCD sẽ đập ngược trở lại)	Đã hoàn thành
Hiện thực được việc bóng bay theo 2 chiều dựa trên phương và chiều của lúc nâng bóng (nếu bóng bay ra rìa LCD sẽ đập ngược trở lại)	Đã hoàn thành

3 BÀI TẬP

Bài tập 1: Thiết kế mô hình phần cứng cần thiết cho yêu cầu trên: cần dùng những phần cứng nào, sử dụng các giao thức giao tiếp gì giữa các phần cứng? **(15% số điểm)**

Để thực hiện trọn vẹn yêu cầu, ta nên sử dụng KIT STM32F4 Discovery (đã được tích hợp đầy đủ tất cả các thành phần cần thiết) để có thể thực hiện yêu cầu với hiệu suất cao nhất. Hình sau chỉ là sơ đồ tham khảo các thành phần phần cứng cần thiết nhất để thực hiện yêu cầu:



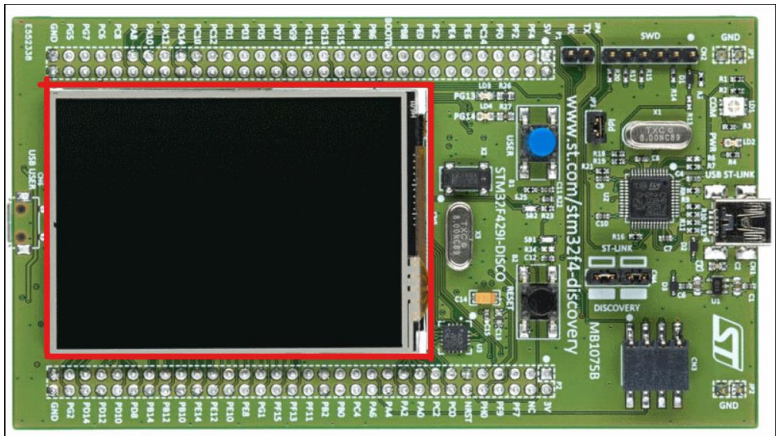
Hình 6 - Các thành phần phần cứng cần thiết

Các thành phần phần cứng được sử dụng:

- ***Vi xử lí STM32F429:*** Vi xử lí STM32F429 được thiết kế dựa trên kiến trúc ARM Cortex-M4, đây là một vi xử lí hiệu suất cao, khả năng xử lí tín hiệu số mạnh mẽ. Đây cũng chính là thành phần phần cứng trung tâm, có chức năng tính toán, xử lí các tác vụ.

Hình 7 - Minh họa vi xử lí STM32F429

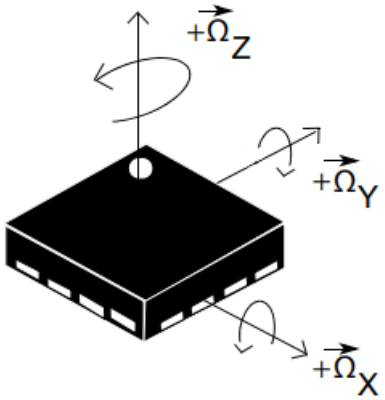
- ***Màn hình LCD****:* Màn hình LCD được tích hợp trong KIT STM32F4 Discovery là màn hình TFT màu, có kích thước là 2.4 inch, là thành phần phần cứng có chức năng hiển thị giao diện chính của trò chơi.



Hình 8 - Màn hình LCD được tích hợp trong KIT STM32F429 Discovery

- ***USB device:*** Kit STM32F429 Discovery thường bao gồm 1 cổng USB, cho phép giao tiếp với máy tính hay các thiết bị khác. Ngoài ra còn hỗ trợ bộ chuyển đổi USB sang Serial giúp dễ dàng giao tiếp qua cổng COM ảo khi kết nối với máy tính. Đây là thành phần phần cứng thực hiện chức năng xuất thông số độ cao khi bóng được tâng ra máy tính qua Virtual Com Port.
- ***Cảm biến Gyroscope:*** được tích hợp trên bản mạch của KIT STM32F4 Discovery, đây là thành phần phần cứng quan trọng, dùng để đo góc

quay, tốc độ góc khi KIT đang chuyển động. Từ những dữ liệu trên ta có thể mô phỏng được chuyển động của bóng trên màn hình LCD cũng như góc độ của tay khi đánh bóng.



Hình 9 - Cảm biến Gyroscope

Các giao thức được sử dụng để giao tiếp giữa các thành phần phần cứng:

- ***Giao thức SPI***: giao thức SPI là một phương pháp giao tiếp phổ biến trong các ứng dụng nhúng, SPI5 là một trong số đó. Trong STM32, các cổng SPI như SPI5 thường được sử dụng để kết nối với các loại cảm biến, màn hình LCD và các thiết bị ngoại vi khác...
- ***Giao thức USB_OTG_HS***: là một tính năng nâng cao cho phép KIT STM32F4 hoạt động như một thiết bị USB (USB device). Cho phép thực hiện việc truyền nhận thông tin với các thiết bị khác như máy tính thông qua cổng USB.

Bài tập 2: Sử dụng RTOS, thiết kế mô hình phần mềm cho yêu cầu trên: nêu công việc của từng task, luồng xử lý dữ liệu như thế nào? **(15% số điểm)**

Project sử dụng 2 task để thực hiện yêu cầu, trong đó 2 task lần lượt có chức năng như sau:

Task 1: Task 1 có chức năng chính là hiển thị giao diện chính của trò chơi trên màn hình LCD và chuyển dữ liệu độ cao sang PC bằng Virtual COM Port. Dữ liệu input của Task 1 được lấy từ hàng được MailQueue gửi từ Task 2 bao gồm: Điểm số (point); Tọa độ tâm bóng (a, b); Bán kính bóng (R); Tín hiệu cờ báo game over (game_over_flag). Từ dữ liệu được lấy từ Task 2, Task 1 thực hiện hiển thị bóng, điểm số và thực hiện kiểm tra game over (nếu có sẽ hiển thị dòng thông báo GAME OVER!). Song song với đó, Task 1 cũng thực hiện gửi tín hiệu độ cao qua PC của người dùng (USB host) thông qua Virtual COM Port.

Bảng 2 - Task 1

```
void StartTask01(void const * argument)
{
 /* init code for USB_DEVICE */
 MX_USB_DEVICE_Init();
 /* USER CODE BEGIN 5 */

 int16_t point;
 int16_t a;
 int16_t b;
 float   R;
 int8_t  game_over_flag;
 uint8_t msg_buf[14] = "Hello Player\n\r";
 uint8_t msg_point[3];
 uint8_t msg_score[7] = "Score: ";
 uint8_t msg_game_over[10] = "GAME OVER!";
 uint8_t TxBuffer[6];
 int16_t total_point = 0;
 uint8_t perfect_hit[10] = "PERFECT!";

 //osSignalWait(0x1, 10000);
 osSignalSet(Task02Handle, 0x1);
 CDC_Transmit_HS(msg_buf, strlen((char *)msg_buf));

 /* Infinite loop */
 for(;;)
 {
        osEvent event = osMailGet(myMailQueueHandle, osWaitForever); //Get mail
        QUEUE_DATA_STRUCT *queue_rx = event.value.p;
        point                       = queue_rx->point;
        a                           = queue_rx->a;
        b                           = queue_rx->b;
        R                           = queue_rx->R;
        game_over_flag  = queue_rx->game_over_flag;

        if(point > 1){
                BSP_LCD_DisplayStringAt(10, 300, perfect_hit, CENTER_MODE);    // display perfect hit
        }
```

```
        osMailFree(myMailQueueHandle, queue_rx); //Free mail queue memory

        sprintf(TxBuffer, "%.1f", R);
        TxBuffer[4] = '\n';
        TxBuffer[5] = '\r';
        CDC_Transmit_HS(TxBuffer, strlen((char *)TxBuffer));  // send R to Virtual COM Port

        // calculate total point
        total_point += point;

        // display coding here
        sprintf(msg_point, "%d", total_point);
        BSP_LCD_DisplayStringAt(0, 0, msg_score, LEFT_MODE);  // display point
        BSP_LCD_DisplayStringAt(100, 0, msg_point, LEFT_MODE);
        BSP_LCD_FillCircle(a, b, R);        // draw filled circle
        if(game_over_flag){
                BSP_LCD_DisplayStringAt(10, 300, msg_game_over, CENTER_MODE);
        }

        // for debug


  osDelay(15);

  //clear to see what does change in LCD
  BSP_LCD_Clear(LCD_COLOR_BLUE);//clear the LCD on blue color

 }
 /* USER CODE END 5 */
}
```

Task 2: Task 2 có chức năng chính là lấy dữ liệu từ cảm biến Gyroscope, sau đó thực hiện xử lí dữ liệu raw sang độ. Từ dữ liệu có được từ Gyroscope, thực hiện giải thuật xử lí để xác định điểm số (point), tọa độ tâm bóng (a, b), bán kính bóng (R), và tín hiệu cờ báo game over (game_over_flag). Sau đó, dữ liệu được gửi qua hàng đợi MailQueue dưới dạng Struct (cụ thể Struct được khai báo là QUEUE_DATA_STRUCT). Trong quá trình này, Task 2 cũng đồng thời thực hiện Blink Led thích hợp ứng với thao tác chơi của người dùng.

Bảng 3 - Task 2

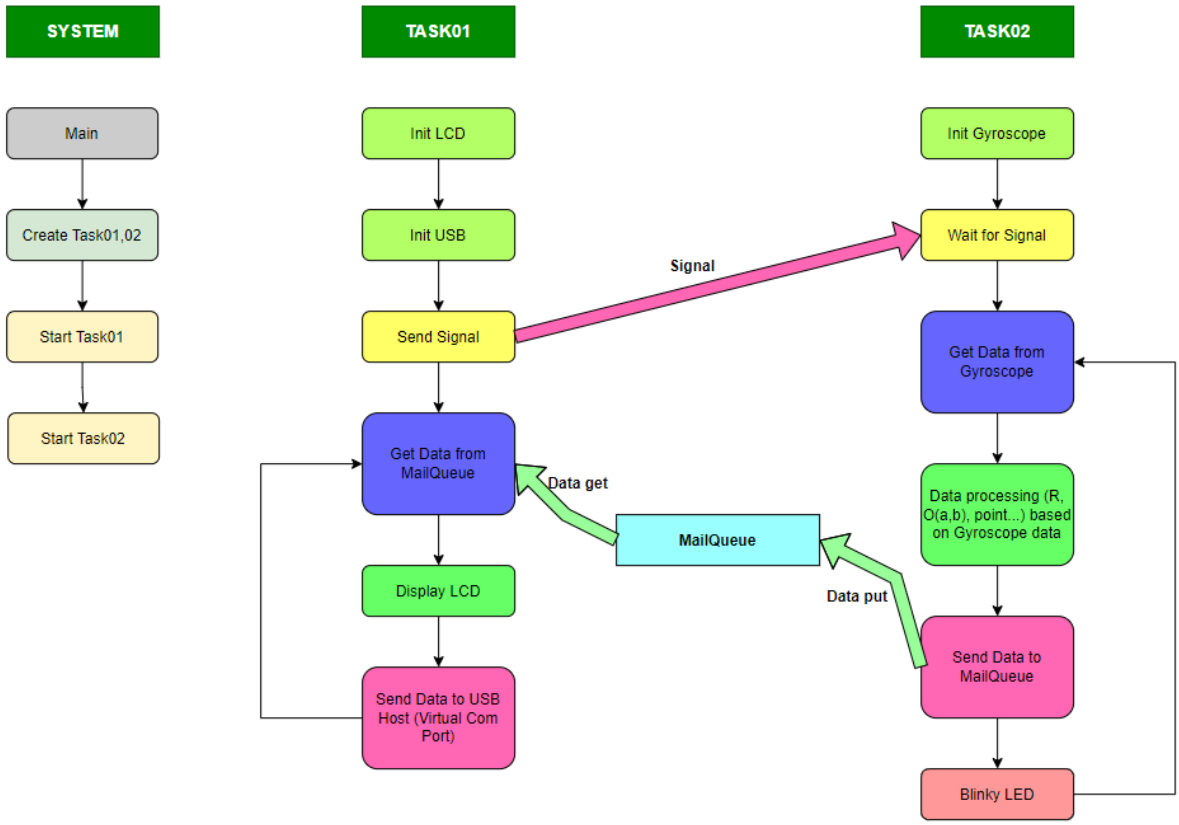
```
void StartTask02(void const * argument)
{
  /* USER CODE BEGIN StartTask02 */
  /* Infinite loop */

    float     gyro_data[3];
    int16_t          point = 0;
    int16_t          a = 120;
    int16_t          b = 160;
    float   R = 15;             //  R_MIN <= R <= R_MAX
    int8_t game_over_flag = 0;
```

```
  float  rate_gyr_x;
  float  rate_gyr_y;
  float  rate_gyr_z;
  float  gyro_angle_x = 0;
  float  gyro_angle_y = 0;
  float  gyro_angle_z = 0;
  float  pre_gyro_angle_x = 0;
  float  pre_gyro_angle_y = 0;
  float  pre_gyro_angle_z = 0;
  float  delt_angle_x = 0;
  float  delt_angle_y = 0;
  float  delt_angle_z = 0;
  float  loop_period = 0.01;
  float  k = 1.01;
  float  R_MAX = 50;
  float  R_MIN = 15;
  float  R_HIT_LEVEL01 = 20;
  float  R_HIT_LEVEL02 = 25;
  float  R_HIT_LEVEL03 = 30;
  float  v_x = 40;
  float  v_y = 40;
  int16_t   THRESHOLD_HIGH = 90;
  int16_t   THRESHOLD_LOW = -90;
  float  WIDTH = 239;
  float  HEIGHT = 319;


  osSignalWait(0x1, osWaitForever); //Wait for signal.

 for(;;)
 {
        BSP_GYRO_GetXYZ(gyro_data);            // get raw data from gyroscope

        // xu li chuyen doi gyroscope raw data sang angle
        rate_gyr_x = (float) gyro_data[0] * L3GD20_SENSITIVITY_500DPS * 0.001;
        rate_gyr_y = (float) gyro_data[1] * L3GD20_SENSITIVITY_500DPS * 0.001;
        rate_gyr_z = (float) gyro_data[2] * L3GD20_SENSITIVITY_500DPS * 0.001;

        gyro_angle_x += rate_gyr_x * loop_period;
        gyro_angle_y += rate_gyr_y * loop_period;
        gyro_angle_z += rate_gyr_z * loop_period;

        gyro_angle_x = (int)gyro_angle_x % 180;
        gyro_angle_y = (int)gyro_angle_y % 180;
        gyro_angle_z = (int)gyro_angle_z % 180;

        delt_angle_x = gyro_angle_x - pre_gyro_angle_x;
        delt_angle_y = gyro_angle_y - pre_gyro_angle_y;
        delt_angle_z = gyro_angle_z - pre_gyro_angle_z;

        pre_gyro_angle_x = gyro_angle_x;
        pre_gyro_angle_y = gyro_angle_y;
        pre_gyro_angle_z = gyro_angle_z;

        // xu li game tai day: xac dinh point, a, b, R(dua vao he so k va gyroscope data), game_over_flag,
blink led
if ((delt_angle_x <= THRESHOLD_LOW || delt_angle_x >= THRESHOLD_HIGH|| delt_angle_y >=
THRESHOLD_HIGH || delt_angle_y <= THRESHOLD_LOW) && game_over_flag == 0 && k < 1){
                //BSP_LCD_FillTriangle(120, 90, 150, 320, 290, 290);          // duoi
                if(R < R_HIT_LEVEL01){
```

```
                k = 1.02;
                point = 2;
        }
        else if(R < R_HIT_LEVEL02){
                k = 1.015;
                point = 1;
        }
        else if(R < R_HIT_LEVEL03){
                k = 1.01;
                point = 1;
        }

        if(k > 1){
                //point = 1;
                HAL_GPIO_WritePin(GPIOG, GPIO_PIN_13, GPIO_PIN_SET);
        }

        ////////////////////////////////////// tinh toan huong bong khi cham vot
        if (delt_angle_x >= THRESHOLD_HIGH){ // cham mat tren
                if (v_y < 0){
                        v_y = -v_y  ;
                }
        }

        if (delt_angle_x <= THRESHOLD_LOW){             // cham mat duoi
                if (v_y > 0){
                        v_y = -v_y;
                }
        }

        if (delt_angle_y >= THRESHOLD_HIGH){ // cham mat trai
                if(v_x < 0){
                        v_x = -v_x;
                }
        }

        if (delt_angle_y <= THRESHOLD_LOW){             // cham mat phai
                if(v_x > 0){
                        v_x = -v_x;
                }
        }

        delt_angle_x = 0;
        delt_angle_y = 0;
}

if(a <= R || a >= WIDTH-R){                     // cham LEFT-RIGHT
        v_x = -v_x;
}

if(b <= R || b >= HEIGHT-R){                    // cham TOP-BOTTOM
        v_y = -v_y;
}

a += (int) (v_x) * 0.025;    // tinh toa do (a~x, b~y)
b += (int) (v_y) * 0.025;

if(a <= R){                     // cham LEFT-RIGHT
        a = R+1;
}
```

```
        if(b <= R){                     // cham TOP-BOTTOM
                b = R+1;
        }

        if(a >= WIDTH-R){               // cham LEFT-RIGHT
                a = WIDTH-R;
        }

        if(b >= HEIGHT-R){              // cham TOP-BOTTOM
                b = HEIGHT-R;
        }

        // kiem tra game over va khoan gioi han cua R
        if(R < R_MIN){
                game_over_flag = 1;
                k = 1;
                HAL_GPIO_WritePin(GPIOG, GPIO_PIN_14, GPIO_PIN_SET);
        }

        if(R >= R_MAX){
                k = 0.99;
        }

        R *= k;

        QUEUE_DATA_STRUCT *queue_tx;                    // declare object to send to queue
        do
        {
                queue_tx = (QUEUE_DATA_STRUCT *)osMailAlloc(myMailQueueHandle,
osWaitForever);
        } while (NULL == queue_tx);

        queue_tx->point         = point; // Packaging data
        queue_tx->a                     = a;
        queue_tx->b                     = b;
        queue_tx->R                     = R;
        queue_tx->game_over_flag = game_over_flag;

        osMailPut(myMailQueueHandle, queue_tx); //Put data into mail queue.

        // for debug
        point = 0;
   osDelay(1);
   HAL_GPIO_WritePin(GPIOG, GPIO_PIN_13, GPIO_PIN_RESET);
  }
  /* USER CODE END StartTask02 */
}
```

Bài tập 3: Thiết kế lưu đồ giải thuật xử lý cho yêu cầu trên? **(30% số điểm)**



Hình 10 - Lưu đồ giải thuật

Giải thích lưu đồ giải thuật:

System: Sẽ thực hiện chạy theo thứ tự:

- Bước vào thực hiện chương trình chính để khởi tạo một số ngoại vi và khai báo, khởi tạo các biến, hằng số cần thiết.
- Thực hiện tạo Task01, Task02 và hàng đợi MailQueue.
- Start Task01 và Start Task02 (2 task này sẽ chạy Concurrency với nhau).

Task01: Thực hiện:

- Khởi tạo LCD để thực hiện chức năng hiển thị.
- Khởi tạo USB để thực hiện xuất thông tin độ cao qua Virtual COM Port tới PC người dùng (USB host).
- Gửi tín hiệu đến Task02 để Task02 bắt đầu thực hiện đoạn chương trình xử lí của nó.
- Lấy dữ liệu từ hàng đợi MailQueue
- Dựa vào dữ liệu lấy từ hàng đợi MailQueue, thực hiện hiển thị bóng, điểm số và thông báo (“GAME OVER!” nếu có).

- Xuất thông tin độ cao qua Virtual COM Port tới PC người dùng (USB host).

Task02: Thực hiện:

- Khởi tạo Gyroscope để thực hiện chức năng đánh bóng.
- Đợi tín hiệu bắt đầu từ Task01.
- Lấy dữ liệu từ Gyroscope và thực hiện xử lí các giải thuật để xác định tâm bóng, bán kính, điểm số và cờ báo game over.
- Các thông số của game được gửi qua hàng đợi MailQueue dưới dạng Struct.
- Thực hiện Blink Led dựa theo hành vi của người chơi.

Bài tập 4: Hiện thực hệ thống và báo cáo kết quả? **(40% số điểm)**

- Hiển thị được giao diện ban đầu:

Bảng 4 - Hiển thị được giao diện ban đầu

Code:

```
// display coding here
        sprintf(msg_point, "%d", total_point);
        BSP_LCD_DisplayStringAt(0, 0, msg_score, LEFT_MODE);  // display point
        BSP_LCD_DisplayStringAt(100, 0, msg_point, LEFT_MODE);
        BSP_LCD_FillCircle(a, b, R);        // draw filled circle
        if(game_over_flag){
                BSP_LCD_DisplayStringAt(10, 300, msg_game_over, CENTER_MODE);
        }
```

Giải thích:

Thực hiện hiển thị như sau:

- Chuyển điểm số thành chuỗi và in ra màn hình.
- Vẽ hình quả bóng dựa vào tọa độ tâm (a, b) và bán kính R.
- Thực hiện in thông báo “GAME OVER!” nếu tín hiệu cờ game over bật.

Hình 11 - Giao diện bắt đầu (trái) và kết thúc(phải)

- Hiện thực được thao tác tâng bóng tại chỗ (điểm giữa màn hình):

Bảng 5 - Hiện thực được thao tác tâng bóng tại chỗ

Code:

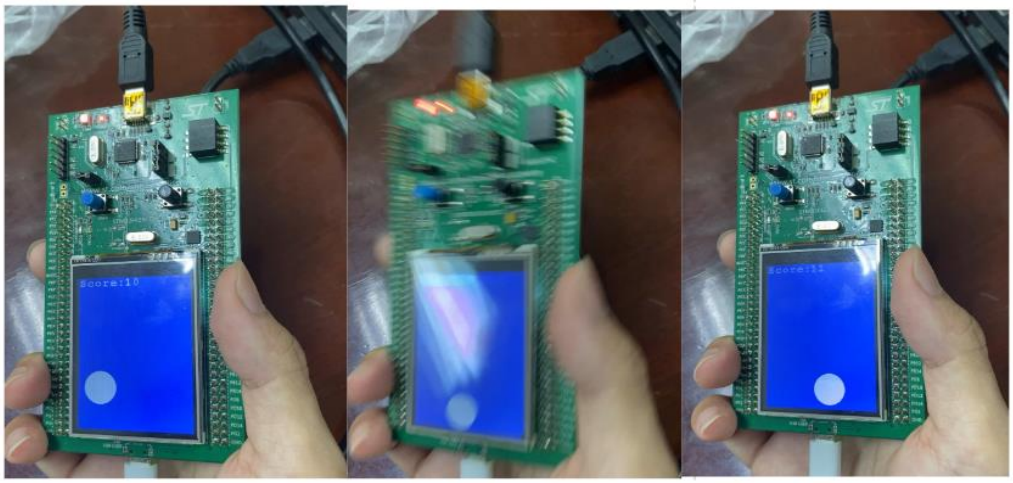
```
// xu li game tai day: xac dinh point, a, b, R(dua vao he so k va gyroscope data),
game_over_flag, blink led
if ((delt_angle_x <= THRESHOLD_LOW || delt_angle_x >= THRESHOLD_HIGH||
delt_angle_y >= THRESHOLD_HIGH || delt_angle_y <= THRESHOLD_LOW) &&
game_over_flag == 0 && k < 1){
                    if(R < R_HIT_LEVEL01){
                            k = 1.02;
                            point = 2;
                    }
                    else if(R < R_HIT_LEVEL02){
                            k = 1.015;
                            point = 1;
                    }
                    else if(R < R_HIT_LEVEL03){
                            k = 1.01;
                            point = 1;
                    }

                    if(k > 1){
                            //point = 1;
                            HAL_GPIO_WritePin(GPIOG, GPIO_PIN_13, GPIO_PIN_SET);
                    }
```

Giải thích:

Ta thực hiện thay đổi độ cao của quả bóng mỗi bằng cách nhân bán kính với một hệ số k, nếu:

- k > 1: Bóng to dần và đang bay lên.
- k < 1: Bóng nhỏ dần và đang rơi xuống.

Nếu người dùng không đánh bóng (tức các giá trị delta của x, y lấy từ Gyroscope không vượt ngưỡng quy định trước) thì hệ số k giữ nguyên và không thực hiện tính điểm, ngược lại thay đổi hệ số k tương ứng với mức độ chính xác khi đánh bóng và tính điểm.



Hình 12 - Thao tác tâng bóng

- Xuất thông số độ cao khi bóng được tâng ra máy tính qua Virtual Com Port:

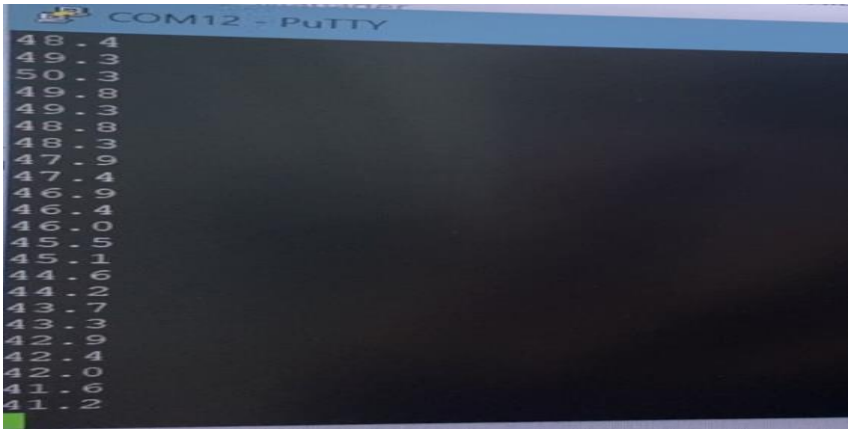
Bảng 6 - Xuất thông số độ cao ra máy tính qua Virtual Com Port

Code:

```
sprintf(TxBuffer, "%.1f", R);
        TxBuffer[4] = '\n';
        TxBuffer[5] = '\r';
        CDC_Transmit_HS(TxBuffer, strlen((char *)TxBuffer));  // send R to Virtual COM Port
```

Giải thích:

Thực hiện chuyển giá trị bán kính R thành chuỗi (ở đây ta coi như giá trị R cũng tương ứng với giá trị độ cao mà quả bóng đang đạt được) sau đó truyền sang PC của người dùng (USB host) thông qua Virtual COM Port sử dụng tốc độ High Speed.



Hình 13 - Thông số độ cao được xuất trên màn hình máy tính

- Hiện thực được chức năng tính điểm:

Bảng 7 - Hiện thực được chức năng tính điểm

Code:

```
total_point += point;
```

Giải thích:

Khi người dùng đánh trúng bóng, điểm số sẽ tự cập nhật một lượng tương ứng với mức độ chính xác khi đánh bóng (1 điểm hoặc 2 điểm đối với perfect hit). Chức năng tính điểm được hiển thị xuyên suốt trên màn hình LCD

Hình 14 - Chức năng tính điểm

- Hiện thực được việc bóng bay theo 1 và 2 chiều dựa trên phương và chiều của lúc nâng bóng (nếu bóng bay ra rìa LCD sẽ đập ngược trở lại):

Bảng 8 - Hiện thực được việc bóng bay theo 1 và 2 chiều

Code:

```
// xu li game tai day: xac dinh point, a, b, R(dua vao he so k va gyroscope data),
game_over_flag, blink led
if ((delt_angle_x <= THRESHOLD_LOW || delt_angle_x >= THRESHOLD_HIGH||
delt_angle_y >= THRESHOLD_HIGH || delt_angle_y <= THRESHOLD_LOW) &&
game_over_flag == 0 && k < 1){
                    //BSP_LCD_FillTriangle(120, 90, 150, 320, 290, 290);          // duoi
                    if(R < R_HIT_LEVEL01){
                              k = 1.02;
                              point = 2;
                    }
                    else if(R < R_HIT_LEVEL02){
                              k = 1.015;
                              point = 1;
                    }
                    else if(R < R_HIT_LEVEL03){
                              k = 1.01;
```

```
                point = 1;
        }

        if(k > 1){
                //point = 1;
                HAL_GPIO_WritePin(GPIOG, GPIO_PIN_13, GPIO_PIN_SET);
        }

        ////////////////////////////////////// tinh toan huong bong khi cham vot
        if (delt_angle_x >= THRESHOLD_HIGH){ // cham mat tren
                if (v_y < 0){
                        v_y = -v_y  ;
                }
        }

        if (delt_angle_x <= THRESHOLD_LOW){             // cham mat duoi
                if (v_y > 0){
                        v_y = -v_y;
                }
        }

        if (delt_angle_y >= THRESHOLD_HIGH){ // cham mat trai
                if(v_x < 0){
                        v_x = -v_x;
                }
        }

        if (delt_angle_y <= THRESHOLD_LOW){             // cham mat phai
                if(v_x > 0){
                        v_x = -v_x;
                }
        }

        delt_angle_x = 0;
        delt_angle_y = 0;
}

if(a <= R || a >= WIDTH-R){                     // cham LEFT-RIGHT
        v_x = -v_x;
}

if(b <= R || b >= HEIGHT-R){                    // cham TOP-BOTTOM
        v_y = -v_y;
}

a += (int) (v_x) * 0.025;    // tinh toa do (a~x, b~y)
b += (int) (v_y) * 0.025;

if(a <= R){                     // cham LEFT-RIGHT
        a = R+1;
}

if(b <= R){                     // cham TOP-BOTTOM
        b = R+1;
}

if(a >= WIDTH-R){                       // cham LEFT-RIGHT
        a = WIDTH-R;
}
```

```
if(b >= HEIGHT-R){                    // cham TOP-BOTTOM
        b = HEIGHT-R;
}
```

Giải thích:
Chiều của bóng được xác định với chiều của vector tổng hợp giữa 2 vector vx và vy:

- Nếu bóng chạm mặt trên-dưới của vợt hoặc của LCD, vy đổi chiều (vy = -vy).
- Nếu bóng chạm mặt trái-phải của vợt hoặc của LCD, vx đổi chiều (vx = -vx).
- Tọa độ tâm bóng (a, b) được cập nhật tương ứng bởi vx, vy nhân cho thời gian của một vòng lặp chương trình (ở đây một vòng lặp khoảng 0.025s)

Các trường hợp vị trí của bóng:

- Tọa độ a của bóng a <= R hoặc a >= WIDTH-R thì bóng chạm cạnh trái-phải của LCD.
- Tọa độ b của bóng b <= R hoặc b >= HEIGHT-R thì bóng chạm cạnh trên-dưới của LCD.

Hình 15 - Tình huống bóng bay ra rìa LCD đập ngược trở lại

Tham khảo:

https://www.keil.com/pack/doc/CMSIS/RTOS2/html/group__CMSIS__RTOS__Message.html

https://hocarm.org/rtos-co-ban-phan-1/

https://hocarm.org/rtos-co-ban-phan-2/

https://www.st.com/resource/en/datasheet/l3gd20.pdf

https://stm32f4-discovery.net/2014/08/library-28-l3gd20-3-axis-gyroscope/

https://itecnotes.com/electrical/converting-raw-gyro-l3gd20h-values-into-angles/